

Data Parallel C++

# Performance, Portability & Productivity

Rakshith Krishnappa, Developer Evangelist @Intel

[rakshith.krishnappa@intel.com](mailto:rakshith.krishnappa@intel.com)

The Intel logo, consisting of the word "intel" in a lowercase, sans-serif font, with a registered trademark symbol (®) to its upper right. The logo is positioned in the bottom left corner of the slide.

intel®

# Performance, Portability and Productivity

01/12/2022	Introduction to Performance, Portability and Productivity for HPC
02/09/2022	Optimization Best Practices using SYCL
03/09/2022	Optimization for Performance Portability across CPU and GPU
03/16/2022	Performance, Portability and Productivity – Mini Hackathon

# What will you learn in this Learning Series

## Session 1 (Jan 12<sup>th</sup> 2022)

- Understand why Performance, Portability and Productivity are important for HPC
- Identify an algorithm and implement using Math Kernel Library and check for performance on CPUs and GPUs
- Implement the same algorithm using basic SYCL programming
- Analyze results using Intel Advisor Roofline and Intel VTune Profiler
- We will do all this on Intel DevCloud on the following CPUs and GPUs:
  - Intel® Xeon® E-2176G Processor with GEN9 GT2 Graphics
  - Intel (R) Core (TM) i9-10920X with Iris XE Max discrete Graphics
  - Intel® Xeon® Gold 6128 Processor
  - Intel® Xeon® Platinum 8153 Processor

# What will you learn in this Learning Series

## Session 2 (Feb 9<sup>th</sup> 2022)

- Use SYCL features to tune the basic algorithm.
- Learn about using ND\_Range kernels and impact of work-group size
- Use private memory and shared local memory to improve performance
- Analyze results using Intel Advisor Roofline and Intel VTune Profiler

# What will you learn in this Learning Series

## Session 3 (Mar 9<sup>th</sup> 2022)

- Optimize algorithm for Performance Portability across CPUs and GPUs
- Analyze results using Intel Advisor Roofline and Intel VTune Profiler and compare all algorithm implementations.
- Understand different accelerator hardware characteristics and further optimize algorithm
- Understand impact of accelerator Occupancy and impact of varying Work-group sizes
- Resources for more advanced tuning using the oneAPI GPU optimization guide

# What will you learn in this Learning Series

## Session 4 (Mar 16<sup>th</sup> 2022)

- Mini-Hackathon and working session, bring your own code, share and ask us any questions.
- We will have a bunch of Intel experts with hardware architecture , SYCL language experts, Performance tuning experts and Tools experts.
- We will have break-out rooms to work with Intel experts.

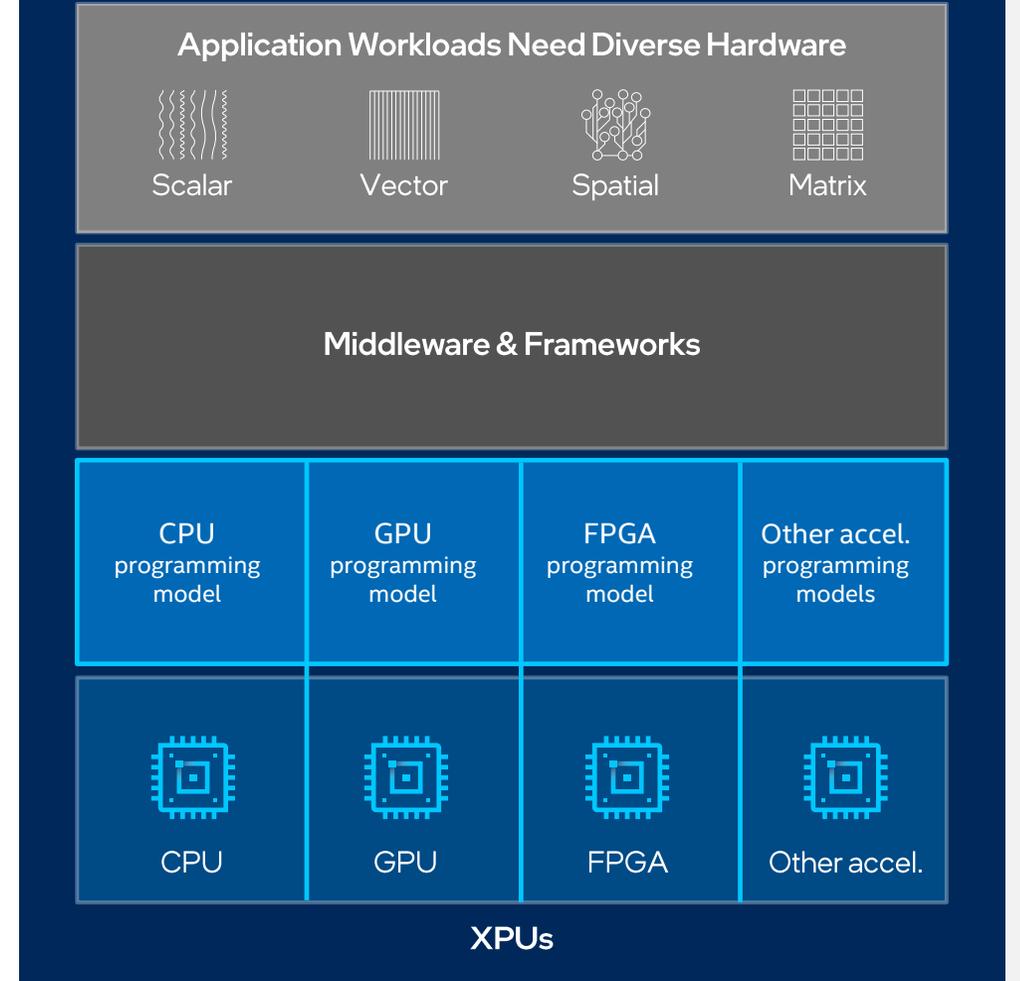
# Programming Challenges for Multiple Architectures

Growth in specialized workloads

Variety of data-centric hardware required

Separate programming models and toolchains for each architecture are required today

Software development complexity limits freedom of architectural choice



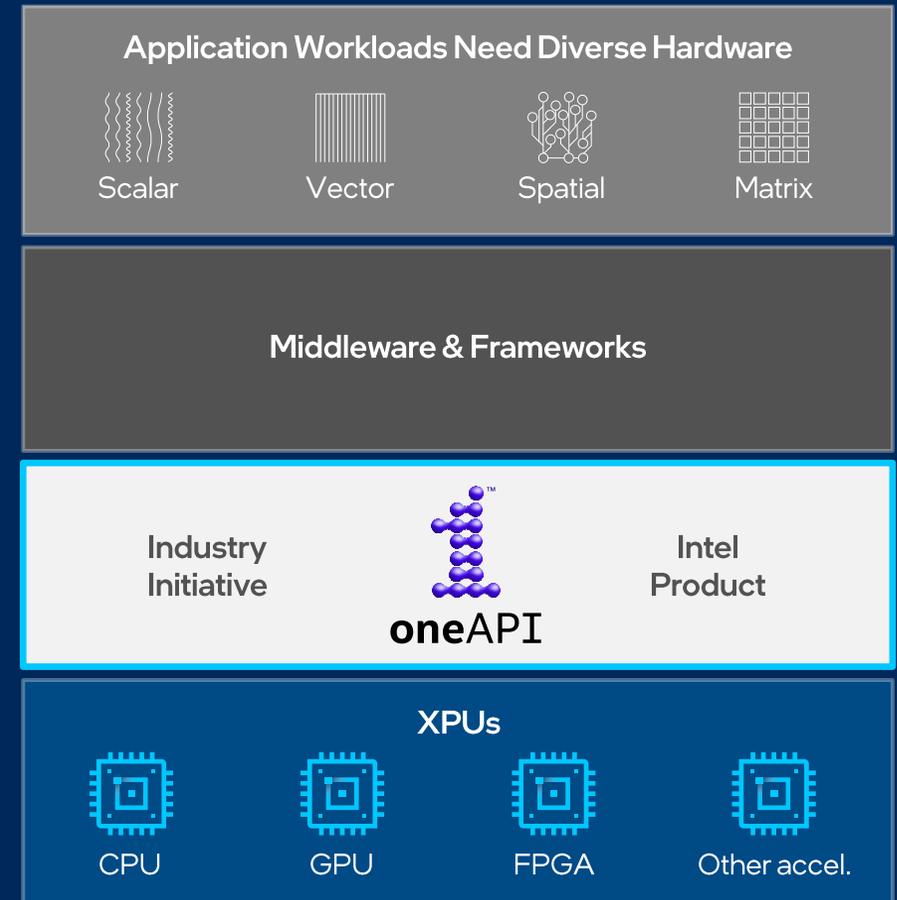
# Introducing oneAPI

Cross-architecture programming that delivers freedom to choose the best hardware

Based on industry standards and open specifications

Exposes cutting-edge performance features of latest hardware

Compatible with existing high-performance languages and programming models including C++, OpenMP, Fortran, and MPI



# Data Parallel C++

Standards-based, Cross-architecture Language  
DPC++ = ISO C++ and Khronos SYCL

## Parallelism, productivity and performance for CPUs and Accelerators

- Delivers accelerated computing by exposing hardware features
- Allows code reuse across hardware targets, while permitting custom tuning for specific accelerators
- Provides an open, cross-industry solution to single architecture proprietary lock-in

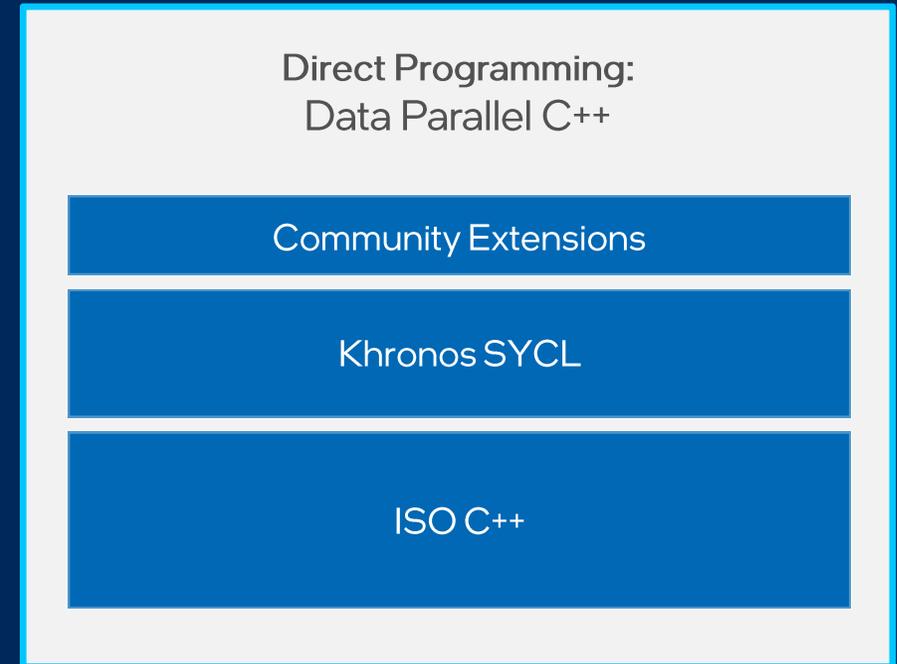
## Based on C++ and SYCL

- Delivers C++ productivity benefits, using common, familiar C and C++ constructs
- Incorporates SYCL from the Khronos Group to support data parallelism and heterogeneous programming

## Community Project to drive language enhancements

- Provides extensions to simplify data parallel programming
- Continues evolution through open and cooperative development

Apply your skills to the next innovation, not rewriting software for the next hardware platform



# Data Parallel C++

DPC++ Essentials Learning Path:

<https://www.intel.com/content/www/us/en/developer/tools/oneapi/training/dpc-essentials.html>

DPC++ Book:

<https://www.apress.com/us/book/9781484255735>

oneAPI GPU Optimization Guide:

<https://www.intel.com/content/www/us/en/develop/documentation/oneapi-gpu-optimization-guide/top/intro.html>

# SYCL 2020

Specification:

<https://www.khronos.org/registry/SYCL/specs/sycl-2020/pdf/sycl-2020.pdf>

# SYCL 2020 Specification



- The SYCL 2020 specification was released on February 9, 2021
- Several major features, including:
  - Unified shared memory
  - Reductions
  - Modern atomics
  - Sub-groups
  - Group algorithms (e.g. reductions, scans)
  - Extension and interoperability mechanisms

# DPC++ Extensions now part of SYCL 2020

Many of DPC++ Extensions became part of SYCL 2020 specification:

- Unified Shared Memory (USM)
- Sub-Groups
- Reductions

DPC++ now adopts the SYCL 2020 syntax for the above features.

# Unified Shared Memory in SYCL 2020

Unified Shared Memory(USM) is now part of SYCL 2020, No changes in USM syntax.

Type	Description	Accessible on Host?	Accessible on Device?
<code>sycl::malloc_device</code>	Allocations in device memory. Programmer must explicitly transfer data between host and device.	No	Yes
<code>sycl::malloc_host</code>	Allocations in host memory. Kernels can access these allocations directly.	Yes	Yes
<code>sycl::malloc_shared</code>	Allocations can migrate between host and device memory. Different implementations may provide different guarantees regarding whether allocations can be accessed by host and device concurrently.	Yes	Yes

# Sub-Groups in SYCL 2020

Sub-Groups are now part of SYCL 2020 Specification

## sub\_group class

A sub-group handle can be obtained from an `nd_item` using `get_sub_group()`

### Before

```
sycl::ONEAPI::sub_group sg = item.get_sub_group();
```

```
sycl::ext::oneapi::sub_group sg = item.get_sub_group();
```

### Now

```
sycl::sub_group sg = item.get_sub_group();
```

```
auto sg = item.get_sub_group();
```

# Sub-Groups in SYCL 2020

DPC++ Sub-Groups Shuffles are now part of SYCL 2020 Sub-Group “Group Algorithms” free functions instead of member functions.

Old DPC++ Compiler	Current DPC++ Compiler (SYCL 2020)
<code>sg.shuffle_down(x, 1);</code>	<code>sycl::shift_group_left(sg, x, 1);</code>
<code>sg.shuffle_up(x, 1);</code>	<code>sycl::shift_group_right(sg, x, 1);</code>
<code>sg.shuffle(x, id);</code>	<code>sycl::select_from_group(sg, x, id);</code>
<code>sg.shuffle_xor(x, mask);</code>	<code>sycl::permute_group_by_xor(sg, x, mask);</code>

# Sub-Groups in SYCL 2020

DPC++ Sub-Groups Collective are now part of SYCL 2020 Sub-Group “Group Algorithms” with function name changes.

Old DPC++ Compiler	Current DPC++ Compiler (SYCL 2020)
<code>broadcast(sg, x, id);</code>	<code>group_broadcast(sg, x, id);</code>
<code>reduce(sg, x, op);</code>	<code>reduce_over_group(sg, x, op);</code>
<code>exclusive_scan(sg, x, op);</code>	<code>exclusive_scan_over_group (sg, x, op);</code>
<code>inclusive_scan(sg, x, op);</code>	<code>inclusive_scan_over_group (sg, x, op);</code>
<code>any_off(sg, x);</code>	<code>any_off_group(sg, x);</code>
<code>all_off(sg, x);</code>	<code>all_off_group(sg, x);</code>
<code>none_off(sg, x);</code>	<code>none_off_group(sg, x);</code>

# Reductions in SYCL 2020

DPC++ introduced a dedicated abstraction for reduction kernels.

This is now part of SYCL 2020.

```
q.parallel_for(nd_range<1>{N, B},  
              Before  
              sycl::ext::oneapi::reduction(sum, 0, sycl::ext::oneapi::plus<>()),  
              [=](nd_item<1> it, auto& tmp) {  
                int i = it.get_global_id(0);  
                tmp += data[i];  
              }).wait();
```

```
q.parallel_for(nd_range<1>{N, B},  
              Now  
              sycl::reduction(sum, 0, sycl::plus<>()), [=](nd_item<1> it, auto& tmp) {  
                int i = it.get_global_id(0);  
                tmp += data[i];  
              }).wait();
```

# Reductions in SYCL 2020

Below example shows changes in reductions in SYCL2020 when using USM and Buffers

	Old DPC++	SYCL 2020
USM	<code>ext::oneapi::reduction(sum, ext::oneapi::plus&lt;&gt;())</code>	<code>sycl::reduction(sum, sycl::plus&lt;&gt;())</code>
Buffers	<code>ext::oneapi::reduction(sum_acc, ext::oneapi::plus&lt;&gt;())</code>	<code>sycl::reduction(sum_buf, h, sycl::plus&lt;&gt;())</code>

# Reductions in SYCL 2020

SYCL 2020 specification extends kernel reductions even further by allowing multiple reductions in a single kernel.

```
auto reduction_min = sycl::reduction(min, sycl::minimum<>());
auto reduction_max = sycl::reduction(max, sycl::maximum<>());

q.parallel_for(nd_range<1>{N, B},
               reduction_min, reduction_max,
               [=](nd_item<1> it, auto& tmp_min , auto& tmp_max) {
    int i = it.get_global_id(0);
    tmp_min.combine(data[i]);
    tmp_max.combine(data[i]);
}).wait();
```

# Intel<sup>®</sup> oneAPI Toolkits Free Availability

## Get Started Quickly

Code Samples, Quick-start Guides, Webinars, Training

[software.intel.com/oneapi](https://software.intel.com/oneapi)

A dark blue navigation menu for Intel DevCloud. It is divided into two main sections. The left section, titled 'Run the tools locally', contains three items: 'Downloads' with a download icon, 'Repositories' with a document icon, and 'Containers' with a container icon. The right section, titled 'Run the tools in the Cloud', features the 'intel DevCloud' logo in white text.

Run the tools locally

Downloads

Repositories

Containers

Run the tools in the Cloud

intel DevCloud

# oneAPI Available on Intel® DevCloud

A development sandbox to develop, test and run workloads across a range of Intel CPUs, GPUs, and FPGAs using Intel's oneAPI software.

## Get Up & Running In Seconds!

[software.intel.com/devcloud/oneapi](https://software.intel.com/devcloud/oneapi)

intel.  
DevCloud



1 Minute to Code

No Hardware Acquisition

No Download, Install or Configuration

Easy Access to Samples & Tutorials

Support for Jupyter Notebooks, Visual Studio Code

# Intel DevCloud

- Login to Intel DevCloud - [devcloud.intel.com/oneapi](https://devcloud.intel.com/oneapi)
- Get Started -> Launch Jupyter Lab option
- Start Terminal and enter command to copy latest content
  - [/data/oneapi\\_workshop/get\\_jupyter\\_notebooks.sh](#)

intel®

# Legal Notices and Disclaimers

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit [www.intel.com/benchmarks](http://www.intel.com/benchmarks).

Performance results are based on testing as of the publication date of the referenced papers and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

**Optimization Notice:** Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

Intel and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.